Continuations in Natural Language

[Extended Abstract]

Chris Barker
University of California, San Diego
9500 Gillman Drive
La Jolla, CA, USA
barker@ucsd.edu

ABSTRACT

Computer scientists, logicians and functional programmers have studied continuations in laboratory settings for years. As a result of that work, continuations are now accepted as an indispensable tool for reasoning about control, order of evaluation, classical versus intuitionistic proof, and more. But all of the applications just mentioned concern *artificial* languages; what about *natural* languages, the languages spoken by humans in their daily life? Do natural languages get by without any of the marvelous control operators provided by continuations, or can we find continuations in the wild? This paper argues yes: that an adequate and complete analysis of natural language must recognize and rely on continuations. In support of this claim, I identify four independent linguistic phenomena for which a simple CPS-based description provides an insightful analysis.

1. INTRODUCTION

The applications of continuations to date are remarkably diverse. As representative examples, I will mention three strands of research here: first, continuations provide an order-independent way of describing evaluation order in formal languages. For instance, Plotkin [10] shows how various Continuation-Passing Style (CPS) transforms can model evaluation disciplines such as call-by-name or call-by-value. Second, Griffin [5], Murthy [9] and others show that continuations are intimately involved in characterizing the computational content of classical (as opposed to intuitionistic) proofs. Third, continuations provide a useful tool for programmers who want to use powerful control operators such as `call/cc` in Scheme (and its analogs in other functional programming languages); conversely, Queinnec [11] shows how to use continuations in order to write in a direct style and still be assured that programs will behave in a reasonable way when it is the program's *users* who have access to powerful control operators (such as the 'back' button on a web browser).

Fourth ACM-SIGPLAN Continuation Workshop '04 Venice, Italy

As diverse as these applications are, they all involve the design and analysis of artificial languages (i.e., programming languages and logical languages), as opposed to natural languages (the sort of languages that humans characteristically use when communicating with each other). We might wonder, then, whether continuations are relevant for the study of natural languages. This is the master question addressed by this paper:

- **[Relevance]** Are there phenomena in natural language that can profitably be analyzed using continuations?

Obviously, I believe that the answer is "yes"! I will try to persuade you to believe likewise by presenting four case studies, each of which has an analysis in terms of continuations.

An affirmative answer to the master question of relevance leads to a number of subquestions, including the following.

- **[Universality]** Even if some natural languages make use of continuation, do all languages make use of continuations?

Languages differ in many ways. For instance, all languages make a distinction between nouns and verbs, but not all languages make a distinction between singular nouns and plural nouns. I have drawn the case studies below exclusively from English, since that simplifies the exposition, but we must remain alert to the possibility that English may be unusual or even unique in the relevant respects. In fact, however, all of the phenomena discussed below have close analogs in many other languages. Coordination in particular is widespread: as far as I know, every language provides some way of saying something equivalent to *John saw Mary and Tom*, where the coordinating conjunction *and* combines two noun phrases (*Mary* and *Tom*) into a single complex noun phrase *Mary and Tom*. If an adequate analysis of coordination depends on continuations, as I argue below, that would strongly suggest that every natural language makes essential use of continuations.

- **[Distribution]** Which specific control operators do natural languages make use of? Which control operators are more 'natural'?

A variety of control operators that make use of continuations have been proposed over the years, some of which differ in fairly subtle ways: `call/cc` versus control ($\mathcal{C}$), `shift` and `reset` versus `fcontrol` and `run`, and so on. Sometimes one

operator can be expressed in terms of another; nevertheless, if one or the other version is more prevalent in natural language, that might suggest that some operators are more 'natural' than others. In the discussion below, I will argue that $\mathcal{C}$ and `fcontrol`/`run` are remarkably well-suited to handling quantification and focus.

- [**Delimitation**] Does natural language prefer delimited or undelimited continuations?

Historically, the first continuation operators were undelimited (e.g., `call/cc` or `J`). Felleisen [4, 3] proposed delimited continuations (sometimes called 'composable' continuations) such as control ('$\mathcal{C}$') and prompt ('%'). Interestingly, the natural-language phenomena discussed in this paper all seem to require some form of delimitation.

2. PRELIMINARIES

The intrepid reader is welcome to skip directly to the case studies, backtracking to this section only if some assumption or technique seems puzzling.

Theoretical computer scientists think deeply about the nature of formal languages. Many of the tools and techniques that are relevant for the study and the design of formal languages apply also to the study of natural language, including lexical analysis, parsing, and denotational semantics. As a result, computer scientists have highly developed intuitions about languages and about good ways to analyze them. These intuitions will go a long way towards understanding the issues and phenomena discussed below; after all, one of my main goals in this paper is to emphasize similarities between formal languages and natural languages.

Nevertheless, there are significant differences between computer science approaches and linguistic approaches. I have made some effort to present a discussion of natural language in a way that makes sense to a computer scientist—but only up to a point. To the extent that the assumptions and techniques agree with standard practice in computer science, well and good; but when there are differences, I will depend on the reader to trust that the linguistic assumptions are coherent and well motivated, and to keep an open mind about what is the "right" way to model language.

2.1 Declarative sentences denote truth values

As a starting point, it is necessary to think of natural language utterances as expressing computations. Natural language is clearly capable of specifying algorithms: just think of a recipe in a cookbook. Natural language can also express imperatives (*Shut the door!*), queries, and can even impinge on the real world in a direct way that formal languages typically do not (*I hereby declare you husband and wife*).

In this paper, however, I will concentrate exclusively on relatively simple and mundane declarative sentences such as *Everyone left*. Declarative sentences will have a type equivalent to a boolean, and their values will be either `true` or `false`. To see how this makes sense, it may help to consider the role of *everyone left* in the conditional sentence *If (everyone left), then shut the door.*

A little thought will reveal that treating declarative sentences as denoting truth values is inadequate in general. After all, there are more than two possible meanings for sentences. Even if *Everyone left* and *The room is empty* are both true, they have different meanings. But the same objection applies for formal languages: '`x == x`' means something very different from '`x == 3`', yet it still makes sense to treat them both as boolean expressions.

In some more elaborate linguistic treatments, sentences denote functions from times and worlds to truth values, with an analogous shift for expressions of other types. In the parlance of linguistics, a treatment in terms of truth values is 'extensional', and a system with times and worlds is 'intensional'. Shan [13] shows that intensionalization can be rendered as a monad. Intensionality is not crucial in any of the discussions below, and the types will be complex enough anyway, so I will use an extensional semantics on which sentences denote truth values.

2.2 Methodology: formal grammar fragments

It is standard in mainstream linguistics when analyzing a particular type of expression to provide a formal grammar approximating the syntax and the semantics of the expressions under study, and that is the approach that I will take in this paper. Although I intend the grammar fragment developed below to capture something genuine about the nature of the natural language expressions, it is important to bear in mind that any formal grammar is at best an approximation of natural language, i.e., a hypothesis, not a definition, and the degree to which the formal treatment accurately reflects the behavior of the natural language expression will always be an empirical issue.

In the fragment, each expression will be assigned membership in some syntactic category, and the name of each syntactic category will also serve to indicate the semantic type of the meanings of expressions in that category.

A concrete example will help make this clear. Here is a direct-style starting point that the fragment below will build on:

(1) Some direct-style lexical entries:

Expression	Syntactic category (= type)	Semantic value
John	`e`	**j**
Mary	`e`	**m**
left	`e→t`	**left**
saw	`e→(e→t)`	**saw**

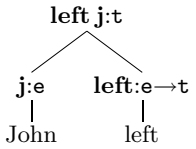
Intuitively, `e` is the category of expressions that denote individuals. Since the semantic job of the proper name *John* is to refer to a particular individual (in this case, the individual **j**), the word *John* is a member of the category `e`. Expressions in a category with a complex label of the form A→B will denote a function from meanings of type A to meanings of type B. Expressions in category `t` denote truth values (booleans), so that an intransitive verb such as *left* has type `e→t` and denotes a function from individuals to truth values.

Syntactic combination in this direct grammar will always correspond semantically to functional application.

(2) fx:B ::= x:A f:A→B

This way of specifying syntactic combination is a kind of BNF notation enhanced in two ways: categories have been labeled with semantic values, and the syntactic categories contain variables over types. The idea is that (2) schematizes over a set of valid BNF rules for any choice of categories A and B. For instance, (2) licenses combining the noun phrase *John* with the intransitive verb phrase *left* (in

that order) by virtue of the following instantiations: A = **e**, B = **t**, f = **left**, and x = **j**. This gives rise to the following syntactic and semantic analysis for the declarative sentence *John left*.



The intended interpretation is that a use of the sentence *John left* will be predicted to be true just in case the function denoted by *left* maps the individual referred to by *John* onto the truth value **true**.

Unlike most formal languages, which tend to prefer a uniform direction of functional application, English allows some arguments to precede the function that applies to them, as in the example immediately above, and some to follow. For instance, in the sentence *John saw Mary*, the transitive verb *saw* combines first with *Mary* on its right, then the verb phrase *saw Mary* combines with the argument *John* to its left. Thus we need a second rule for syntactic combination to allow for arguments that appear on the right:

(3) fx:B ::= f:A→B x:A

One of the ways in which the formal grammar developed here is unrealistic is that the syntactic categories of *left* and *saw* as given in (1) do not specify which arguments follow and which precede. Thus in addition to deriving *John left* and *John saw Mary*, it also incorrectly derives the ungrammatical sentences **Left John* and **Saw John Mary* (the star indicates that the string it is prefixed to is not well-formed).

It is not difficult to build grammars that take linear order into account; but since order will not play an important role in what follows, I have opted to ignore linear order in an effort to keep the correspondence between syntactic category and semantic type as transparent as possible: since the syntactic category of *saw* is **e→(e→t)**, it is clear that the semantic type of its value will be a function from individuals to functions from individuals to truth values.

2.3 Strategy: case studies

My strategy will be to discuss a number of cases in which a continuation-based analysis is elegant and potentially insightful. I will discuss several different cases rather than developing a single analysis in more depth for two reasons: first, because which solutions strike a responsive cord varies from one person to another; and second, for the sheer joy of considering a number of different natural language phenomena.

There will be four case studies: quantification, coordination, focus particles, and misplaced modifiers. (I will introduce each one of these phenomena using concrete examples below, of course.) The treatments of quantification and coordination have been developed in some detail in [1], but the proposals for focus particles and misplaced modifiers are new to this paper. There are a few other continuation-based analyses of natural language already in the literature: Shan [15] proposes a continuation-based analysis of question formation, and Shan and Barker [17] discuss a continuation approach to the phenomena of weak crossover and superiority, which unfortunately requires background discussions that are too complex to attempt here. In addition, at this conference, Shan will present a new continuation-based analysis of negative polarity.

3. CASE STUDY: QUANTIFICATION

The most carefully worked out natural-language application of continuations to date concerns quantification ([2], [1], [17]). According to the simple analysis above in section 2, the sentence *John left* denotes the truth value (**left j**) and does not involve quantification; a sentence like *Everyone left*, however, is quantificational, and means roughly $\forall x.\mathbf{left}\, x$. (We say that the symbol '∀' is a quantifier, and likewise '∃' below.)[1]

The main problem of interest is what I call 'scope displacement': even when a quantificational expression such as *everyone* occurs in a deeply embedded position, the quantifier it contributes can take semantic scope over the entire sentence.

(4) a. John (saw everyone).
b. $\forall x.\mathbf{saw}\, x\, \mathbf{j}$

Here, even though *everyone* is buried within the verb phrase *saw everyone*,[2] in the paraphrase in (4b), the logical quantifier ∀ takes scope over the entire meaning. Continuations, of course, are superb at allowing a deeply embedded operator to take control over a larger expression in which it is embedded, and the analysis described here is based on that ability.

3.1 From quantification to continuization

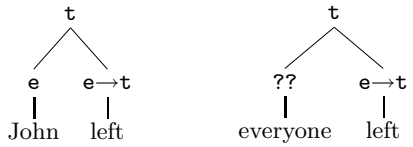
We proceed by deducing a type for quantificational expressions like *everyone*, *no one*, or *someone*. Syntactically, these quantificational expressions can be substituted in any position occupied by a proper name. Thus since *John saw Mary* is grammatical, the sentences *Everyone saw Mary*, *John saw everyone*, *Someone saw everyone*, etc., will all be grammatical as well. Yet it is problematic assigning *everyone* type **e**, the same type as *John* or *Mary*, since there is no particular individual who has all and only the properties that are true of everyone. The inadequacy of supposing that quantificational noun phrases denote individuals is even more stark in the case of *no one*: if *No one left* is true, which individual has the property of leaving?

We can solve this dilemma by considering the parallelism in the following two syntactic analyses:

[1] I use expressions in the first-order predicate calculus to specify what I have in mind for the meaning. At one level, this suggests that natural language can be translated into some suitable logical language, and this is in fact a perfectly legitimate technique. At a deeper level of analysis, however, I am assuming for the purposes of this paper that words are logical constants denoting some specific individual, truth value, or function, and that the denotations of complex phrases are composed from the denotations of their constituent words entirely through functional application as governed by the various syntactic combination rules as given. In Montague's [8] terms, the intermediate logical description language is non-essential, and could be dispensed with entirely if desired.

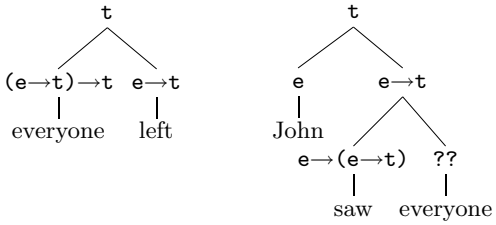
[2] The parentheses in (4a) indicate what I take to be a constituent. There is abundant evidence that the transitive verb *saw* forms a constituent with its direct object *everyone* to the exclusion of the subject *John*. To give just one argument, note that verb phrases can be coordinated (see section 5): *John (saw everyone) and (called home)*.

(5)



If *John* has a type that combines with the type of *left* to form a complex expression of type `t`, then presumably *everyone* must also have a type that combines with the type of *left* to produce an expression of type `t`. Since we want to avoid giving *everyone* the type `e`, the next simplest type that will serve is `(e→t)→t`. This is in fact the main insight proposed by Montague [8]: that noun phrases such as *everyone* and *no one* can have the type `(e→t)→t`, which he called a *generalized quantifier* (for reasons I won't discuss here). Intuitively, what this analysis says is that *everyone* is a function mapping a verb phrase to a truth value.

(6)



This naive solution works out fine for the example *Everyone left* (although it has the rather disturbing effect of reversing the direction of functional application, a point I'll return to shortly). But if we attempt to place a quantificational expression in some other position where a proper name can occur (i.e., instead of *John saw Mary* we have *John saw everyone*, as in the diagram on the right of (6)), the solution breaks down: the generalized quantifier type `(e→t)→t` does not suffice, and we seem to need a yet higher type.

This is the point at which continuations enter the picture. Consider again the diagrams in (5) from the point of view of continuations. What is the continuation of the expression *John* relative to the sentence *John left*? It will be a function mapping individuals into truth values, i.e., a function of type `e→t`. Not coincidentally, this happens to also be the type of the verb phrase *left*. From this perspective, the proposed generalized quantifier type for *everyone*, `(e→t)→t`, is a function from its continuation (type `e→t`) to a truth value.

Now we can understand better how to deal with the situation depicted in (6). What is the continuation of *everyone* in the sentence *John saw everyone*? Well, once again it will be a function mapping individuals to truth values. More specifically, it will be the function mapping each individual x to `true` just in case John saw x. If only we had a way of providing the denotation of *everyone* with access to its continuation, a single type would serve in all of the examples considered so far.

In operational terms, we need the control operator $\mathcal{C}$:

(7) $E[\mathcal{C}M] \triangleright M(\lambda x.E[x])$

What (7) says is that when $\mathcal{C}M$ occurs in an evaluation context E, evaluation proceeds by packaging the context as a continuation (i.e., $\lambda x.E[x]$), then replacing the entire computation with M applied to the continuation.

Then if *everyone* denotes $\mathcal{C}(\lambda P.\forall x.Px)$, *John saw everyone* will denote $(\lambda P.\forall x.Px)(\lambda x.\mathbf{saw}\,x\,\mathbf{j})$, which is equivalent via β-reduction to $\forall x.\mathbf{saw}\,x\,\mathbf{j}$, as desired.

My denotational implementation of this analysis will be based on the following CPS transform:

(8) a. $\overline{\alpha} = \lambda\kappa.\kappa\alpha$ (for any constant α)
b. $\overline{MN} = \lambda\kappa.\overline{M}(\lambda m.\overline{N}(\lambda n.\kappa(mn)))$

I prove a simulation result for this transform in [1].

Note that the transform does not provide a rule for expressions of the form $\lambda x.M$. It is possible to omit such a rule here because the natural language constructions considered in this paper involve only functional application, and never lambda abstraction.[3]

As a consequence, the type of the CPS transform of an expression of any direct type X will be $(X\to\sigma)\to\sigma$, where σ is some answer type. In particular, the type of the CPS transform of a direct expression of type A→B will simply be $((A\to B)\to\sigma)\to\sigma$, rather than the more traditional (e.g., [7]) type $A'\to((B'\to\sigma)\to\sigma)$, where X' is the type of the transform of an expression of type X. This difference in types accounts for the expression '$\kappa(mn)$' in (8b) rather than the traditional '$mn\kappa$'. I find this typing much simpler; however, it seems to be wrenching for people used to the standard transform, so this is one of the places where I must ask the reader to keep an open mind about the right way to do things.

The CPS transform of the lexical items in (1) all depend on the rule governing constants in (8a):

(9) CPS transforms of the lexical entries in (1):

John	`(e→A)→A`	$\lambda\kappa.\kappa\,\mathbf{j}$
Mary	`(e→A)→A`	$\lambda\kappa.\kappa\,\mathbf{m}$
left	`((e→t)→A)→A`	$\lambda\kappa.\kappa\,\mathbf{left}$
saw	`((e→(e→t))→A)→A`	$\lambda\kappa.\kappa\,\mathbf{saw}$

Thus for the purposes of the transform, *left* counts as a constant, even though its direct denotation **left** is a function of type `e→t`.

Here A is a variable over syntactic categories (in effect, over types). For the sake of quantification alone, we could replace the type variables here with `t`. The more general rules will be needed, however, when we extend the fragment to handle focus in the next section.

The syntactic combination rules for the CPS grammar will be as follows:

(10) $\lambda\kappa.\overline{\mathrm{M}}(\lambda m.\overline{\mathrm{N}}(\lambda n.\kappa(mn))):(\mathrm{B}\to\mathrm{E})\to\mathrm{D}$
::= $\overline{\mathrm{M}}:((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D}$ $\overline{\mathrm{N}}:(\mathrm{A}\to\mathrm{E})\to\mathrm{C}$

(11) $\lambda\kappa.\overline{\mathrm{N}}(\lambda n.\overline{\mathrm{M}}(\lambda m.\kappa(mn))):(\mathrm{B}\to\mathrm{C})\to\mathrm{E}$
::= $\overline{\mathrm{N}}:(\mathrm{A}\to\mathrm{D})\to\mathrm{E}$ $\overline{\mathrm{M}}:((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D}$

These syntactic combination rules just make explicit how to syntactically and semantically combine expressions that have already undergone the CPS transform. Thus (10) is the CPS analog of (3), and (11) is the analog of (2).

Some of the complexity in (10) and (11) could be reduced if we chose C = D = E. This would suffice for present purposes except for the treatment of focus in the next section, which depends on allowing a control operator to return an answer that is not of type `t`.

[3] It is an interesting question whether this abstraction-free approach can be maintained in a more complete analysis of natural language. One strategy is to use continuations instead of lambda abstraction whenever natural language seems to require lambda abstraction; see, e.g., [17] for a continuation-based treatment of relative clause formation.

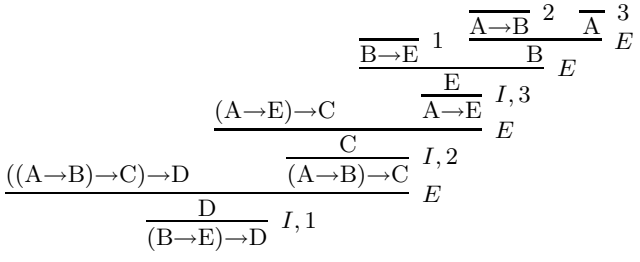
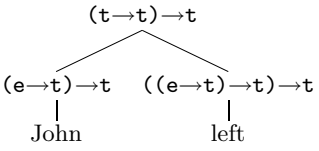


Figure 1: Derivation of (10): a natural-deduction proof using only → Elimination and → Introduction that ((A→B)→C)→D, (A→E)→C ⊢ (B→E)→D.

The proof in figure 1 may help untangle the types. The semantic values in (10) (and hence the application rule in the CPS transform given in (8b)) are just the Curry-Howard labelling of the proof in figure 1. An analogous derivation is available for (11), but is not included here.

These rules give us the following derivation:



$$[\![\textit{John left}]\!] = \lambda\kappa.[\lambda\kappa.\kappa\mathbf{left}](\lambda f.[\lambda\kappa.\kappa\mathbf{j}](\lambda x.\kappa(fx))) \rightsquigarrow \kappa.\kappa(\mathbf{left}\,\mathbf{j})$$

which, after application to the trivial continuation $\lambda x.x$, yields the same denotation provided by the direct grammar, namely, **left j**.

At this point we can add lexical entries for some quantificational noun phrases that have no direct counterpart.

(12) everyone `(e→t)→t` $\lambda\kappa.\forall x.\kappa x$
someone `(e→t)→t` $\lambda\kappa.\exists x.\kappa x$

Since *everyone* has the same type as the CPS transform of *John*, we now have

$$[\![\textit{Everyone left}]\!] = \lambda\kappa.[\lambda\kappa.\kappa\mathbf{left}](\lambda f.[\lambda\kappa.\forall x.\kappa x](\lambda x.\kappa(fx))) \rightsquigarrow \lambda\kappa.\forall x.\kappa(\mathbf{left}\,x),$$

as well as

$$[\![\textit{John saw everyone}]\!] \rightsquigarrow \lambda\kappa.\forall x.\kappa(\mathbf{saw}\,x\,\mathbf{j})$$

which yield $\forall x.\mathbf{left}\,x$ and $\forall x.\mathbf{saw}\,x\,\mathbf{j}$ after application to the trivial continuation, as desired.

Note that continuization has restored the natural direction of functional application (which had been reversed by the first strategy depicted in (6)). That is, not only is the direction of (the CPS analog of) functional application in *John left* right-to-left, the direction of application in the continuized grammar for handling *Everyone left* is also right-to-left.

3.2 Quantificational ambiguity

The transform as specified in (8b) imposes a left-to-right evaluation discipline. But we might just as well have used instead

(13) $\overline{MN} = \lambda\kappa.\overline{N}(\lambda n.\overline{M}(\lambda m.\kappa(mn)))$

in which the right-hand component, $\overline{\mathrm{N}}$, is evaluated first. For sentences containing only one quantificational noun phrase, order of evaluation makes no difference. But for sentences containing two or more quantificational noun phrases, a difference in order of evaluation corresponds to a difference in the predicted meaning:

(14) a. Someone saw everyone.
b. $\exists x\forall y.\mathbf{saw}\,x\,y$
c. $\forall y\exists x.\mathbf{saw}\,x\,y$

Given the availability of both evaluation orders, the sentence in (14a) containing two quantificational noun phrases will have meanings that (after application to the trivial continuation) reduce to either (14b) (for left-to-right evaluation) or (14c) (for right-to-left evaluation).

The consensus in the field is that sentences like (14a) are in fact ambiguous in exactly the way considered here. The interpretation in (14b) corresponds to a situation in which there is some particular person who has the property of seeing everyone, and the interpretation in (14c) corresponds to a situation in which there is a potentially different see-er who saw each person.

Quantifier scope ambiguity has been studied in considerable detail. For more discussion of the predictions of continuation-based analyses with respect to this type of ambiguity, see [1], [2] and [17].

4. CASE STUDY: FOCUS

In this section I propose that a phenomenon called 'focus', along with so-called focus particles such as *only*, behave like Sitaram's [18, 19] `run` and `fcontrol`. One point of interest is that these control operators rely on delimited continuations. Delimited continuations have received a considerable amount of attention recently, and in fact are the topic of at least three papers at the workshop. I believe that they play an important role in natural language as well. In fact, I would be (mildly) surprised to find a natural language operator that behaved exclusively in an undelimited manner.

4.1 `fcontrol` and `run`

There are many control operators that rely on delimited continuations, including Felleisen's control and prompt, shift and reset, and others. I will discuss Sitaram's `fcontrol` and `run` here for the simple reason that they provide exactly the functionality required for describing the behavior of focus particles such as *only*.

Roughly, `fcontrol` provides a way to throw a signal, and `run` catches and handles signals thrown by `fcontrol`.

The `fcontrol` operator takes one argument, and throws two values: the argument to `fcontrol` and the prefix of the continuation of the `fcontrol` expression up to the closest enclosing `run`.

The `run` operator takes two arguments: an expression possibly containing an occurrence of `fcontrol`, and a "handler" function that processes the two arguments thrown by `fcontrol`.

For instance, in

```
(+ 1 (run (* 2 (+ (fcontrol 3) 4))
          (lambda (x k) (k (k x)))))
```

The invocation of `fcontrol` throws the two values 3 and a (truncated) continuation κ. Since κ is the continuation of the expression "`(fcontrol 3)`" relative to the larger expression only up to the closest occurrence of `run`, κ will be the function $\lambda y.(*\,2\,(+\,y\,4))$. The handler procedure will bind `x` to 3 and `k` to κ, and the result of the entire computation will be 37.

The continuation caught by the `fcontrol` is delimited, in the sense that it does not contain any material outside the scope of the enclosing `run`; in particular, the function corresponding to the continuation does not contain the increment operation (there is no "+ 1" inside of κ). In this example, since the continuation is just a function, it can be called several times without replacing the current evaluation context. This allows expressions like `(k (k x))` as in the example at hand, in which one occurrence of a continuation takes an expression involving another occurrence of the same continuation as an argument. This is the sense in which delimited continuations are 'composable'.

Before we can make use of `run` and `fcontrol`, we must consider the meaning of sentences containing focus marking and focus particles.

4.2 Focus

Most (probably all) languages provide some way of marking some constituent in a sentence as having extra prominence. In spoken English, this is typically accomplished in part by a local maximum in the fundamental frequency (the lowest frequency at which the vocal folds are vibrating). By convention, the location of such a 'pitch accent' is indicated typographically by setting the most affected word in capital letters:

(15) a. JOHN saw Mary.
b. John SAW Mary.
c. John saw MARY.

There is a distinct but elusive difference in meaning among these sentences that depends on the location of the pitch accent. In each case, it remains true that John saw Mary, but which piece of information is being emphasized differs. In traditional terms, the constituent containing the pitch accents is said to be 'in focus', which means (very roughly) that it carries the new information provided by the sentence.

These observations can be sharpened by noting that the location of the pitch accent correlates with the precise piece of information requested by a question.

(16) a. Who saw Mary?
b. What did John do to Mary?
c. Who did John see?

Thus (15a) can be a suitable answer only to the question in (16a), and not to either (16b) or (16c), and similarly for the other answer/question pairs.

The semantic effect of the location of pitch accent becomes even more tangible in the presence of a focus particle such as *only*, *also*, or *too*.

(17) a. John only drinks PERrier.
b. John only DRINKS Perrier.

We say that *only* 'associates' with whatever element is in focus. With a pitch accent on the first syllable of *Perrier*, then, *Perrier* will be in focus, and (17a) conveys at least the information paraphrased in (18):

(18) a. John drinks Perrier.
b. There is nothing else that John drinks other than Perrier.

The particle *only*, then, picks out some element in the situation and contrasts it with other possible alternative choices: John doesn't drink whiskey, he doesn't drink milk, he only drinks Perrier.

With a pitch accent on the verb *drinks* in (17b), however, the appropriate paraphrases differ:

(19) a. John drinks Perrier.
b. There is nothing else that John does with Perrier other than drink it.

It remains true that John drinks Perrier, but now the element of the situation that *only* puts into contrast with other alternatives is the activity of drinking. According to (17b), all John does with Perrier is drink it: he doesn't sell it, he doesn't photograph it, he doesn't bathe in it.

Note that the conditions under which (17a) and (17b) will be true are mutually distinct: (17a) can be true even if John sometimes bathes in Perrier, and (17b) can be true even if John sometimes drinks whiskey. Thus in the presence of *only*, the location of the pitch accent can determine whether a sentence is true or false.

I will treat pitch accent in a sentence as if it were an operator F immediately preceding the constituent that is in focus (i.e., that bears the pitch accent). I will continue to use capitals to help guide pronunciation.

(20) a. John only drinks F(PERrier).
b. John only F(DRINKS) Perrier.

Now we're ready to attempt an analysis in terms of `run` and `fcontrol`. The key is to ask the following question: given pitch accent on *Perrier*, what precisely is the relation that holds between John and Perrier and nothing else?

(21) a. John only drinks F(PERrier).
b. $\lambda xy.\mathbf{drink}\, x\, y$

The answer is the relation that holds between John and some object x if John drinks x. In other words, (21b) is a continuation, namely, the continuation of the focussed expression delimited by the enclosing *only*.

Considering next the contrasting example with pitch instead on *drinks*, what is the relation that holds between John and the activity of drinking and nothing else?

(22) a. John only F(DRINKS) Perrier.
b. $\lambda xy.x\ \mathbf{Perrier}\ y$

The answer this time is the relation that holds between John and some activity x if John does x to Perrier. Once again, the desired relation is the continuation of the focussed word up to but not including *only*.

Based on these examples, we can now guess that the semantic effect of pitch accent on a constituent is exactly `fcontrol`.[4] So where I wrote 'F' in (22), the meaning is `fcontrol`. (In a happy coincidence, in this application we can construe the 'f' of `fcontrol` as mnemonic for 'focus'.) Similarly, the meaning of *only* will invoke `run`. It remains only to specify the handler routine that unpacks the information provided by the use of `fcontrol`:

$$[\![only\ P]\!] = \texttt{run} P \lambda x \kappa y.(\mathbf{and}\ (\kappa x y)\ (\forall z(\mathbf{or}(\mathbf{equal}\, x\, z)(\mathbf{not}(\kappa z y)))))$$

This denotation gives rise to the following analyses:

(23) John only drinks F(PERrier).
(**and** (**drinks Perrier j**)
(∀ *z* (**or** (**equal Perrier** *z*)
(**not** (**drinks** *z* **j**)))))

(24) John only F(DRINKS) Perrier.
(**and** (**drinks Perrier j**)
(∀ *z* (**or** (**equal drinks** *z*)
(**not** (*z* **Perrier j**)))))

[4] Chung-chieh Shan first noticed the similarity between my analysis of focus and Sitaram's operators.

These meanings are equivalent to the paraphrases we started with in (18) and (19).

Now, it only makes sense to bother building continuations if the meaning captured by the continuation can be arbitrarily complex. The examples we have discussed so far have been as simple as possible, so it is worth considering examples in which the delimited continuation is more complicated.

(25) Mary only tried to dance with F(JOHN).
(and (tried-to-dance-with j m)
(∀ z (or (equal j z)
(not (tried-to-dance-with z m)))))

(26) Mary only tried to F(DANCE) with John.
(and (tried-to-dance-with j m)
(∀ z (or (equal dance z)
(not (tried-to-z-with j m)))))

The simple denotational fragment in section 8 does not attempt to reconstruct the full power of `run` and `fcontrol`, but it does handle the examples discussed here.[5]

5. CASE STUDY: COORDINATION

One of the distinctive features of natural language is the pervasive use of polymorphic coordination.

(27) a. [John left] and [Mary left]. `t`
b. John [left] and [slept]. `e→t`
c. John [saw] and [remembered] Mary. `e→(e→t)`
d. [John] and [Mary] left. `e`

The types in the right column of (27) correspond to the (direct, pre-CPS) type of the bracketed expressions coordinated by *and*. In (27a), two clauses (type `t`) coordinate to form a complex sentence; in (27b), two verb phrases (type `e→t`) coordinate to form a complex verb phrase; and so on.

What about quantificational noun phrases? They also can coordinate:

(28) a. Someone or everyone left.
b. John or everyone left.

Even more interesting, they can more or less freely coordinate with proper names as in (28b), providing confirmation that proper names and quantificational noun phrases are syntactically interchangeable, as predicted by the analysis in section 3.

We can arrive at a continuation-based analysis of coordination if we consider that (27d) means the same thing as *John left and Mary left*.[6] The continuation of the coordinated phrase is $\lambda x.\mathbf{left}\ x$; what the conjunction does is take this continuation and distribute it across each of the conjuncts. The resulting analysis of conjunction involves adding two new rules for syntactic combination:

(29) $\lambda\kappa.\mathbf{and}(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa)$:A ::= $\overline{\mathrm{L}}$:A "and" $\overline{\mathrm{R}}$:A

(30) $\lambda\kappa.\mathbf{or}(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa)$:A ::= $\overline{\mathrm{L}}$:A "or" $\overline{\mathrm{R}}$:A

These rules differ from one another only in substitution of *or* for *and*.

The syntactic parts of these rules simply say that anywhere that an expression of type A can occur, an expression of the form "A_1 and A_2" or "A_1 or A_2" can also occur, as long as A_1 and A_2 are themselves expressions of category A.

What the semantic parts of the rules say is: whatever you were planning on doing to the value provided by the expression in this position, first do it to the value of the left conjunct, then do it to the value of the right conjunct, and conjoin the two results. This schema guarantees the following example paraphrases:

(31) a. John left and slept. John left and John slept.
b. John and Mary left. John left and Mary left.
c. John or everyone left. John left or everyone left.

One sign of the utility of coordination is that in Wall Street Journal text, *and* is the second most commonly used word (first place goes to *the*). If suitably constrained with syntactic marking (such as parentheses) overtly marking the syntactic strings involved, coordination as a control operator could provide a rather appealing programming device. Imagine being able to write `if (x == (2 or 3)) then ...` and have it mean `if ((x == 2) or (x == 3)) then ...`.[7]

6. CASE STUDY: MISPLACED MODIFIERS

As a final example of a natural language construction that might profit from a continuation-based analysis, consider the following data:

(32) a. An occasional sailor walked by.
b. John drank a quiet cup of tea.

The modifier *occasional* is misplaced:[8] there is no specific sailor (nor even any set of sailors) that has the property of being occasional; rather, it is the event of a sailor walking by that happens occasionally compared with other relevant events. Similarly, in (32b), it is not the cup of tea that is quiet in the relevant sense, but the activity of drinking the tea.

Once again, we have an embedded element that needs to take semantic force at a higher level, and once again we can allow that expression to take control by providing it with access to its continuation. (Chris Potts (personal communication) first suggested using continuations to analyze misplaced modifiers, though he shouldn't be held responsible for the shortcomings of my treatment here.) Assuming we know what the adverb *occasionally* means, we can give the misplaced adjective *occasional* the following denotation: $\mathcal{C}\lambda\kappa.\mathbf{occasionally}(\kappa\lambda x.x)$, and similarly for *quietly*

[5]There is an important dependence on the context of utterance that goes unrecognized in this analysis. If *John only DRANK Perrier* quantified over absolutely everything that John might have done with Perrier, then it would entail that he didn't buy Perrier, that he didn't swallow Perrier, that he didn't taste Perrier, that he didn't raise Perrier to his lips, that he didn't do all kinds of things to Perrier that he must have done. The standard assumption is that the quantification over alternatives must be context-dependent, so that what a use of *only* really means is that there is no other *contextually-relevant* thing that John did to Perrier. How precisely to calculate what ought to count as a contextually-relevant activity is a problem for AI, and not for linguistics.

[6]There are other kinds of coordination not treated here. For instance, *John and Mary are a happy couple* cannot be paraphrased as **John is a happy couple and Mary is a happy couple*. Interestingly, unlike conjunction, disjunction displays only the kind of behavior characterized by the rule given in (30).

[7]Hayo Thielecke points out that J, an APL-like language whose web site (`http://www.jsoftware.com`) claims that it uses "constructs and syntax which closely mirror those of natural language", has a construction that is a special case of the operator imagined here. More specifically, under certain circumstances `x (f g h) y` evaluates as `(x f y) g (x h y)`, which is isomorphic to example (27c).

[8]If you prefer fancy terminology, this is a type of hypallage.

and *quiet*. The idea is to replace the misplaced adjective with the trivial adjective meaning $\lambda x.x$, and then let its adverbial counterpart take scope over the complete sentence.

This analysis gives the following equivalences:

(33) a. An occasional sailor walked by.
Occasionally, a sailor walked by.

b. John drank a quiet cup of tea.
Quietly, John drank a cup of tea.

Additional details (such as the type of adjectives and nouns) are given in the cumulative fragment below in section 8.

7. HISTORICAL NOTES

If natural language is rife with phenomena that beg for a continuation-based analysis, why hasn't anyone noticed before? The answer, of course, is that some people *have* noticed. In fact, I would suggest that in addition to the many independent discoveries of continuations described by Reynolds [12], Richard Montague [8] also invented a technique that relies in a limited way on continuations.

Montague was a logician at UCLA who was one of the major figures in the early days of establishing formal approaches to natural language semantics. In 1970 he constructed the first popular formal analysis of quantification [8]. He did this by suggesting that quantificational noun phrases such as *everyone* differ from non-quantificational noun phrases such as *John* in just the way I proposed in section 3: quantificational noun phrases are in effect control operators whose meanings are properly expressed as functions on their own continuations.

The main limitation of Montague's approach is that noun phrases were the only type of expression that had access to their continuations; nevertheless, with hindsight, anyone familiar with continuation-passing style programming will immediately recognize a primitive form of continuation passing in Montague's formal grammars. Joe Goguen, my colleague at UCSD, was a colleague of Montague's at UCLA during the 70's, and he tells me that the connection between Montague's techniques and continuations was noticed long ago, at least in the folklore, if not in the literature.

It has only been recently, however, that people have explored this connection systematically, providing a more general mechanism for accessing continuations in natural language analyses. In particular, Herman Hendriks' 1993 dissertation [6] proposes a type-shifting system that in effect produces CPS transforms as needed (and may deserve to be counted as yet another independent invention of a continuation-based system). Philippe de Groote [2] provides an analysis of simple quantification based on the $\lambda\mu$-calculus, and a paper by me [1] explores a continuation-based approach to quantification in considerable detail. Since then, Chung-chieh Shan has proposed a number of continuation-based analyses of natural language phenomena ([14], [15], [16]), including his paper at this conference.

Incidentally, if natural languages do make abundant use of continuations, then Reynolds was doubly right to speak of the 'discovery' of continuations rather than their 'invention'. It is intriguing to consider the possibility that that the presence of continuation-based control operators in the native language of the various discoverers may even have inspired their proposals at some subconscious level.

8. CUMULATIVE FRAGMENT WITH RESET

Figure 2 gives a CPS grammar describing a fragment of English with a context-free syntax and a denotational semantics. The fragment includes versions of the four analyses discussed above for quantification, focus, coordination, and misplaced modifiers.

In addition, the fragment provides a reset operator, associated here with the complementizer *that* (as discussed immediately below). This section will argue for all of the four phenomena that the relevant continuations must at least sometimes be delimited.

The meanings listed in the examples below are guaranteed to be equivalent to the denotations provided by the fragment up to β-reduction and application to the trivial continuation $\lambda x.x$.

For instance, here are some simple examples involving zero, one, and two quantificational noun phrases:

(34) a. John left. $\quad$ **left j**
b. Everyone left. $\quad \forall x.\mathbf{left}\ x$
c. John saw Mary. $\quad$ **saw m j**
d. John saw everyone. $\quad \forall x.\mathbf{saw}\ x\ \mathbf{j}$
e. Someone saw everyone. $\quad \exists x.\forall y.\mathbf{saw}\ y\ x$

Since the fragment does not include extra combination rules for right-to-left evaluation, there is only one interpretation for *Someone saw everyone* (see section 3.2 for discussion).

I now introduce embedded clauses and a reset operator.

(35) a. John claimed [Mary left]. $\quad \mathbf{claim}\ (\mathbf{left}\ \mathbf{m})\ \mathbf{j}$
b. John claimed [everyone left]. $\quad \forall x.\mathbf{claimed}\ (\mathbf{left}\ x)\ \mathbf{j}$
c. John claimed that [everyone left]. $\quad \mathbf{claimed}\ (\forall x.\mathbf{left}\ x)\ \mathbf{j}$

Unlike *saw*, which denotes a relation between two individuals, *claimed* relates an individual and a proposition. Syntactically, *claimed* takes a clause as its first argument. Thus the bracketed strings in (35) are all complete clauses in their own right.

The interpretation in (35b) says that John made several different claims, one for each person. The interpretation in (35c) says that he made one single claim, a claim about everybody. Both interpretations seem to be valid.

In continuation terms, we need to be able to delimit the continuation for *everyone* so as to extend only as far as the material in the embedded clause. In order to experiment with delimitation, I will adopt the following expository strategy: as shown by comparing (35b) with (35c), *claimed* optionally allows its argument clause to be introduced by the complementizer *that*. By giving *that* the semantics of a reset operator, we can add or subtract delimitation at will and see what happens. As shown in (35c), for instance, the presence of the *that* delimits the embedded clause and produces the second legitimate interpretation.

The following set of sentences illustrates the analysis of focus.

left	$((\mathtt{e}\to\mathtt{t})\to A)\to A$	$\lambda\kappa.\kappa$ **left**
saw	$((\mathtt{e}\to(\mathtt{e}\to\mathtt{t}))\to A)\to A$	$\lambda\kappa.\kappa$ **saw**
claimed	$((\mathtt{t}\to(\mathtt{e}\to\mathtt{t}))\to A)\to A$	$\lambda\kappa.\kappa$ **claimed**
John	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa$ **j**
Mary	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa$ **m**
everyone	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\forall x.\kappa x$
someone	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\exists x.\kappa x$
a	$(((\mathtt{e}\to\mathtt{t})\to\mathtt{e})\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa$ **a**
sailor	$((\mathtt{e}\to\mathtt{t})\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa$ **sailor**
tall	$(((\mathtt{e}\to\mathtt{t})\to(\mathtt{e}\to\mathtt{t}))\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa$ **tall**
occasional	$(((\mathtt{e}\to\mathtt{t})\to(\mathtt{e}\to\mathtt{t}))\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.$**occasionally**$(\kappa(\lambda x.x))$

$\lambda\kappa.\overline{\mathrm{M}}(\lambda m.\overline{\mathrm{N}}(\lambda n.\kappa(mn))):(\mathrm{B}\to\mathrm{E})\to\mathrm{D}$	::=	$\overline{\mathrm{M}}:((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D}$	$\overline{\mathrm{N}}:(\mathrm{A}\to\mathrm{E})\to\mathrm{C}$	
$\lambda\kappa.\overline{\mathrm{N}}(\lambda n.\overline{\mathrm{M}}(\lambda m.\kappa(mn))):(\mathrm{B}\to\mathrm{C})\to\mathrm{E}$	::=	$\overline{\mathrm{N}}:(\mathrm{A}\to\mathrm{D})\to\mathrm{E}$	$\overline{\mathrm{M}}:((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D}$	
$\lambda\kappa.$**and**$(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa):\mathrm{A}$	::=	$\overline{\mathrm{L}}:\mathrm{A}$	"and"	$\overline{\mathrm{R}}:\mathrm{A}$
$\lambda\kappa.$**or**$(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa):\mathrm{A}$	::=	$\overline{\mathrm{L}}:\mathrm{A}$	"or"	$\overline{\mathrm{R}}:\mathrm{A}$
$\lambda\kappa.\langle\overline{\mathrm{X}},\kappa\rangle:\mathrm{A}\to((\mathrm{A}\to\mathtt{t})\times\mathrm{A})$	::=	"F"	$\overline{\mathrm{X}}:\mathrm{A}\to\mathtt{t}$	
$\lambda\overline{\mathrm{X}}\kappa.$**only**$(\overline{\mathrm{X}}\kappa):\mathrm{A}\to\mathtt{t}$	::=	"only"	$\overline{\mathrm{X}}:\mathrm{A}\to((\mathrm{B}\to\mathtt{t})\times\mathrm{B})$	
$\lambda\kappa.\kappa(\overline{\mathrm{X}}(\lambda x.x)):\mathrm{A}$	::=	"that"	$\overline{\mathrm{X}}:\mathrm{A}$	

Figure 2: A grammar in continuation passing style covering most of the examples discussed in the text.

(36) a. John only saw F (MARY).
only$\langle\lambda\kappa(\kappa\mathbf{m}),\lambda x.$**saw** $x\,\mathbf{j}\rangle$

b. John only F (SAW) Mary.
only$\langle\lambda\kappa.\kappa$ **saw**$,\lambda x.x\,\mathbf{m}\,\mathbf{j}\rangle$

c. John only F (SAW MARY).
only$\langle\lambda\kappa.\kappa($**saw m**$),\lambda x.x\mathbf{j}\rangle$

d. John only claimed F (MARY LEFT).
only$\langle\lambda\kappa.\kappa($**left m**$),\lambda x.$**claimed** $x\,\mathbf{j}\rangle$

e. John claimed Mary only saw F (TOM).
***only**$\langle\kappa.\kappa\mathbf{t},\lambda x.$**claimed** $($**saw** $x\,\mathbf{m})\,\mathbf{j}\rangle$

f. John claimed that Mary only saw F (TOM).
claimed $($**only**$\langle\kappa.\kappa\mathbf{t},\lambda x.$**saw** $x\,\mathbf{m}\rangle)\,\mathbf{j}$

Here **only** is a function taking an ordered pair $\langle\overline{\mathrm{X}},\kappa\rangle$ and returning `true` just in case $\overline{\mathrm{X}}\kappa$ is true and there is no other (contextually relevant) meaning $\overline{\mathrm{Y}}$ of the same type as $\overline{\mathrm{X}}$ such that $\overline{\mathrm{Y}}\kappa$ is true. Thus the interpretation for *John only saw MARY* given (36a) entails that John saw Mary and that there is no other individual that John saw.

Examples (36a) and (36b) show that the pitch accent marker F can take arguments of different type (here, a transitive verb *saw* rather than a noun phrase *Mary*).

Examples (36b), (36c), and (36d) show that the focus marker can take either a single word (e.g., *SAW*), a complex phrase (*SAW MARY*), or even an entire clause (*MARY LEFT*) as its argument. (In the last case, the only thing that John claimed was the Mary left; he did not claim that Tom left, he did not claim that Mary called, etc.)

Comparing (36e) with (36f), we see that once again embedded clauses must be delimited in order to arrive at the correct interpretation: (36e) does not mean that Tom was the only person that John claimed Mary saw; rather, it means that John claimed that Tom was the only person Mary saw. Thus the presence of the reset operator provided by *that* in (36f) is crucial to correctly delimiting the scope of *only*.

The next pair of examples shows the interaction of focus with quantificational noun phrases.

(37) a. John only F(SAW) someone.
only$\langle\lambda\kappa.\kappa$ **saw**$,\lambda y.\exists x.y\,x\,\mathbf{j}\rangle$

b. John only F (SAW SOMEONE).
only$\langle\lambda\kappa.\exists x.\kappa($**saw** $x),\lambda x.x\,\mathbf{j}\rangle$

c. John only saw F (SOMEone).
only$\langle\lambda\kappa.\exists x.\kappa\,x,\lambda x.$**saw** $x\,\mathbf{j}\rangle$

Because the focus marker F takes a continuized meaning as its argument, it is able to handle quantificational foci with no trouble.

The next examples demonstrate the analysis of coordination.

(38) a. John left and Mary left. **and**$($**left j**$)($**left m**$)$

b. John and Mary left. **and**$($**left j**$)($**left m**$)$

c. John saw Mary and left. **and**$($**saw m j**$)($**left j**$)$

d. John saw Mary or everyone. **or**$($**saw m j**$)(\forall x.$**saw** $x\,\mathbf{j})$

Note that (38d) involves coordinating a proper name with a quantificational noun phrase, which works fine.

(39) a. John claimed Mary or Tom left.
or$($**claimed** $($**left m**$)\,\mathbf{j})($**claimed** $($**left t**$)\,\mathbf{j})$

b. John claimed that Mary or Tom left.
claimed$($**or**$($**left m**$)($**left t**$))\,\mathbf{j}$

Examples (39a) and (39b) show the behavior of coordination with and without a reset at the level of the embedded clause. In (39a), John either claims Mary left or claims Tom left; in (39b), John makes a single indeterminate claim that either Mary or Tom left. Since the interpretation in (39b) is certainly possible for this sentence, it provides additional evidence that a reset for delimiting embedded clauses must be available.

Finally, we have misplaced modifiers:

(40) a. John saw a tall sailor.
saw (**a**(**tall sailor**)) **j**

b. John saw an occasional sailor.
occasionally (**saw** (**a sailor**) **j**)

c. John claimed that an occasional sailor left.
claimed (**occasionally** (**left** (**a sailor**))) **j**

The adjective *tall* is a normal adjective and has no special control properties. The adjective *occasional*, however, takes scope over an entire clause. In (40c), in order for the leaving to be occasional rather than the claiming activity to be occasional, it is once again necessary to provide a reset delimiting the scope of the misplaced modifier within the embedded clause.

The examples above show that quantification, focus, coordination, and misplaced modifiers all require some form of delimitation at least some of the time in order to generate appropriate interpretations.

A word of warning: each of the analyses in this brief survey has grave inadequacies in the simple form presented here. In addition, many of the interactions among the phenomena behave badly in the cumulative fragment (consider, for instance, that is not possible to generate an analysis for pitch accent on just one conjunct, even thought this is something that native speakers have no trouble interpreting, as for *John only claimed that Mary or TOM left*). This is the normal state of affairs when dealing with natural language, of course; as Sapir put it, "all grammars leak". The delight comes from figuring out an equally simple grammar that performs better. In any case, I am not aware of any potential problem that could not be dealt with by a suitably-developed continuation-based analysis along the lines of the approximations offered here; in other words, I believe each of these proposals to be a viable approach.

9. CONCLUSIONS

I have suggested that continuations provide an appealing analysis of a variety of natural language phenomena. It is possible for a skeptic to claim that natural language semantics has gotten by well enough without continuations so far. To be sure, each of the phenomena discussed above have well-established treatments in the linguistic literature that do not mention continuations. In the same way, computer scientists were perfectly able to deal meaningfully with the difference between call-by-value versus call-by-name before Plotkin's CPS analysis [10]. Yet I take it that these days everyone recognizes that a full understanding of evaluation disciplines requires continuations (or else something very like continuations). It is my hope, then, that the examples in this paper, or other analyses yet to be discovered, either individually or cumulatively, will someday make continuations seem as indispensable for the description of natural language as they currently are for the theory of computation and logic.

In closing, I would like to add one more subquestion to the list of questions discussed in the introductory section.

- [**Innovation**] Are there uses for continuations in natural language that computer scientists haven't thought up yet?

Besides whatever intrinsic interest there might be in finding continuations in a natural setting, it is conceivable that once we look more closely, natural language will do interesting things with continuations that have not yet been dreamt up by theoretical computer scientists. I doubt that any of the natural language constructions treated above will seem breathtakingly new to an experienced continuation hacker; but then, I have chosen these examples precisely in order to maximize the degree to which they seem like garden-variety control operators. In the same spirit that pharmaceutical companies survey compounds harvested from tropical rain forests in hopes of finding medically useful substances unknown to laboratory scientists, we should consider that there are a lot of poorly understood languages out there—who knows what amazing control operators lurk in languages spoken in the forests of New Guinea?

10. ACKNOWLEDGMENTS

Thanks to Harry Mairson, Hayo Thielecke and Chung-chieh Shan.

11. REFERENCES

[1] C. Barker. Continuations and the nature of quantification. *Natural Language Semantics*, 10:211–242, 2002.

[2] P. de Groote. Type raising, continuations, and classical logic. In van Rooy and Stokhof [20], pages 97–101.

[3] M. Felleisen. *The Calculi of λ_v-CS Conversion: A Syntactic Theory of Control and State in Imperative Higher-Order Programming Languages*. PhD thesis, Indiana University, Aug. 1987. Also as Tech. Rep. 226, Department of Computer Science, Indiana University.

[4] M. Felleisen. The theory and practice of first-class prompts. In *POPL '88: Conference Record of the Annual ACM Symposium on Principles of Programming Languages*, pages 180–190, New York, 1988. ACM Press.

[5] T. G. Griffin. A formulae-as-types notion of control. In *POPL '90: Conference Record of the Annual ACM Symposium on Principles of Programming Languages*, pages 47–58, New York, 1990. ACM Press.

[6] H. Hendriks. *Studied Flexibility: Categories and Types in Syntax and Semantics*. PhD thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 1993.

[7] A. R. Meyer and M. Wand. Continuation semantics in typed lambda-calculi (summary). In R. Parikh, editor, *Logics of Programs*, number 193 in Lecture Notes in Computer Science, pages 219–224, Berlin, 1985. Springer-Verlag.

[8] R. Montague. The proper treatment of quantification in ordinary English. In R. Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*, pages 247–270. Yale University Press, New Haven, 1974.

[9] C. R. Murthy. *Extracting Constructive Content from Classical Proofs*. PhD thesis, Department of Computer Science, Cornell University, Aug. 1990. Also as Tech. Rep. TR90-1151.

[10] G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1(2):125–159, 1975.

[11] C. Queinnec. Inverting back the inversion of control or, continuations versus page-centric programming. Rapport de Recherche LIP6 2001/007, Laboratoire d'Informatique de Paris 6, 2001.

[12] J. C. Reynolds. The discoveries of continuations. *Lisp and Symbolic Computation*, 6(3–4):233–247, 1993.

[13] C.-c. Shan. Monads for natural language semantics. In K. Striegnitz, editor, *Proceedings of the ESSLLI-2001 Student Session*, pages 285–298, Helsinki, 2001. 13th European Summer School in Logic, Language and Information.

[14] C.-c. Shan. A variable-free dynamic semantics. In van Rooy and Stokhof [20], pages 204–209.

[15] C.-c. Shan. A continuation semantics of interrogatives that accounts for Baker's ambiguity. In B. Jackson, editor, *SALT XII: Semantics and Linguistic Theory*, pages 246–265, Ithaca, 2002. Cornell University Press.

[16] C.-c. Shan. Quantifier strengths predict scopal possibilities of Mandarin Chinese *wh*-indefinites. Draft manuscript, 2003.

[17] C.-c. Shan and C. Barker. Explaining crossover and superiority as left-to-right evaluation. Draft manuscript, 2003.

[18] D. Sitaram. Handling control. In *PLDI '93: Proceedings of the ACM Conference on Programming Language Design and Implementation*, volume 28(6) of *ACM SIGPLAN Notices*, pages 147–155, New York, June 1993. ACM Press.

[19] D. Sitaram and M. Felleisen. Control delimiters and their hierarchies. *Lisp and Symbolic Computation*, 3(1):67–99, Jan. 1990.

[20] R. van Rooy and M. Stokhof, editors. *Proceedings of the 13th Amsterdam Colloquium*. Institute for Logic, Language and Computation, Universiteit van Amsterdam, 2001.